Probabilistic Capacity and Optimal Coding for Asynchronous Channel

Ning Cai
The State Key Lab. of ISN
Xidian University,
Xi'an, Shaanxi, 710071, China
Email: caining@mail.xidian.edu.cn

Siu-Wai Ho
Dept. of Electrical Engineering
Princeton University
Princeton, NJ 08544, U.S.A.
Email: siuho@princeton.edu

Raymond W. Yeung
Dept. of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
Email: whyeung@ie.cuhk.edu.hk

***Abstract*— The continuous-time *asynchronous channel* as a model for time jitter in a communication system with no common clock between the transmitter and the receiver was introduced in [1]. The paper unveiled that it is not necessary for the receiver clock to re-synchronize with the transmitter clock within a fixed maximum time in order to achieve reliable communication. In this paper, the runlength limited code is shown not to be optimal for this channel model. An upper bound on the rate loss is given for the constraint that the transmitter and receiver clocks cannot be out of synchronization for a fixed maximum time. The probabilistic capacity of the asynchronous channel is determined and the result is used to design an optimal code. The relation between coding in constrained channels and random number generation is also discussed.**

I. Introduction

The effect of time jitter at the physical level was captured by a continuous-time model instead of a discrete-time model in [1]. The paper introduced an asynchronous channel model not only for the tracking problem arising in information storage systems, but also for any continuous-time communication system with no common clock between the transmitter and the receiver. At least in the absence of noise, it is actually not necessary to impose an upper bound on the time to re-synchronize the transmitter and the receiver clocks in order to achieve reliable communication. Moreover, the paper introduced an optimal code which can be used to determine the combinatorial capacity of the asynchronous channel. These results are briefly reviewed in Section II before runlength limited (RLL) codes are shown to be suboptimal for the asynchronous channel through an example. Then we obtain an upper bound on the rate loss when we impose the constraint that the transmitter and receiver clocks cannot be *out-of-sync* for a fixed maximum time. In Section III, we determine the capacity of the asynchronous channel by a probabilistic approach. Based on this result, an asymptotically optimal code is designed in Section IV. Our results also establish a connection between the code design for constrained channels and random number generation.

II. Rate Loss for a Maximum Runlength Constraint

We assume that $S(t)$ is a step function which takes values from a finite input alphabet $\mathcal{X} = \{0, 1, 2, \ldots, p-1\}$, where it is assumed that $p \geq 2$ to avoid triviality. An interval on which $S(t)$ takes a constant value is called a *run*, and the lengths of runs are called *runlengths*. The step functions are sent through a class of asynchronous channels called the $[\mathcal{A}, \xi]$ channel, where $\mathcal{A}$ is a set of positive real numbers, and ξ is a real number at least equal to 1. We assume that there exists a minimum element in $\mathcal{A}$ denoted by d. In a communication session with finite duration, the channel takes as its inputs step functions whose runlengths except possibly for the last run are elements of the set $\mathcal{A}$, called the *runlength set*. The last run is called the *incomplete run* if its length is not in $\mathcal{A}$. The parameter ξ, called the *jitter ratio*, will be explained in the next paragraph. We will characterize optimal self-synchronizable codes for the $[\mathcal{A}, \xi]$ channel.

Consider an asynchronous channel that distorts the input signal in the time domain as follows: a run of length l in the input is reproduced in the output as a run of length between al and bl, where $0 < a < b$. It is a basic observation that a run of length l and a run of length l', where $l < l'$, are always distinguishable in the output if and only if $(b-\epsilon)l < al'$ for any $\epsilon > 0$, or equivalently, $bl \leq al'$. Therefore, we are motivated to define an important parameter $\xi = \frac{b}{a}$, called the *jitter ratio*, so that $bl \leq al'$ becomes $\xi l \leq l'$.

Let $T > 0$, where T may be infinity, be the duration of a communication session. Consider step functions defined on the interval $[0, T]$ satisfying the following properties:

1) The duration is equal to t, where $t \leq T$.
2) The runs take values from the input alphabet $\mathcal{X}$.
3) The lengths of the runs, except for possibly the last run, take values in the runlength set $\mathcal{A}$.
4) If the length of the last run is not in $\mathcal{A}$, then it is less than d.

A set of step functions satisfying the above conditions is called a *code*, and the step functions are called *codewords* of the code. The class of all such codes is denoted by $\mathcal{C}_L$.

We assume that the receiver can always recognize the end of the communication session from the output signal. As an example, in a magnetic recording system, if the communication session spans the length of the tape, then its end can automatically be detected when the tape is finished during playback.

A code $\mathcal{S}$ with domain $[0,T]$ is zero-error if and only if for all $S(t), S'(t) \in \mathcal{S}$, one of the following holds:

(i) $S(t)$ and $S'(t)$ have different numbers of runs;

(ii) $S(t)$ and $S'(t)$ have the same number of runs and there exists a run for which they take different values in $\mathcal{X}$;

(iii) $S(t)$ and $S'(t)$ have the same number of runs and take the same values in all the runs, and there exists an index i such that $\xi l_i \leq l'_i$ or $\xi l'_i \leq l_i$, where l_i and l'_i are the lengths of the ith run in $S(t)$ and $S'(t)$, respectively.

We refer to an asynchronous channel with runlength set $\mathcal{A}$ and jitter ratio ξ as the $[\mathcal{A}, \xi]$ channel, and refer to a zero-error code in $\mathcal{C}_L$ as an $[\mathcal{A}, \xi]$ code. An optimal $[\mathcal{A}, \xi]$ code in $\mathcal{C}_L$ is a zero-error code which contains the largest number of codewords. To obtain an optimal code, we first construct a list $\mathcal{L}(\mathcal{A}, \xi) = \{l_1^*, l_2^*, l_3^*, \dots, \}$ of runlengths by "Construction $(\mathcal{A}, \xi)$" below recursively.

Construction $(\mathcal{A}, \xi)$**:**

Step 1 Let l_1^* be the smallest element d of the set $\mathcal{A}$.

Step j Having taken $l_1^*, l_2^*, l_3^*, \dots, l_{j-1}^*$, take l_j^* to be the smallest element of the closed set $\mathcal{A} \cap [\xi l_{j-1}^*, \infty)$.

Note that the list $\mathcal{L}(\mathcal{A}, \xi)$ so constructed is finite or countably infinite depends on whether the procedure stops in a finite number of steps. Let Ψ be the subclass of the code $\mathcal{C}_L$ such that all the runlengths in a codeowrd are in $\mathcal{L}(\mathcal{A}, \xi)$. We have shown in [1] that Ψ is an optimal code in $\mathcal{C}_L$. This implies that an RLL code, which imposes an upper limit on the runlengths, is not optimal in general. We will see this in Example 2.

Example 1: For $\mathcal{A} = [a, \infty)$, $\mathcal{L}(\mathcal{A}, \xi) = \{a\xi^t : t = 0, 1, 2, \dots\}$.

Example 2: For $\mathcal{A} = \{a, a+1, \dots, \}$, where a is a positive integer, the members of the list $\mathcal{L}(\mathcal{A}, \xi)$ are given recursively by

$$l_1^* = a$$

$$l_j^* = \lceil \xi l_{j-1}^* \rceil, \ j = 2, 3, \dots .$$

Let $p = 2$, and let the logarithms be in the base 2 so that the capacities are expressed in bits per unit time.

i) Let $a = 1$ and $\xi = 7.1/6$. Consider achieving zero-error communication by an RLL code. Such a code is characterized by two parameters (d+1) and (k+1), which are the minimum and maximum runlengths, respectively. Here, an RLL code attaining the maximum rate is the $(0, 5)$ code, which is explained as follows. First, for $0 \leq d \leq 5$, the maximum k that can be taken is 5 because $(5+1) \cdot \frac{7.1}{6} > 7$, which means that if the runlength 6 is used, then the runlength 7 cannot be used. Thus for $0 \leq d \leq 5$, we should let $d = 0$ and $k = 5$ in order to maximize the runlength set. For $d > 5$, since $(d+1) \cdot \frac{7.1}{6} > d+1$, the runlength $d+1$ cannot be used and therefore k must be equal to d. In other words, only the runlength d can be used, and such a code is obviously inferior to the $(5, 5)$ code, which in turn is inferior to the $(0, 5)$ code. Hence, the $(0, 5)$ code, with runlength set $\{1, 2, \dots, 6\}$, is the optimal RLL code that can be employed for zero-error communication.

Alternatively, an $[\mathcal{A}, \xi]$ code with runlength set $\mathcal{L}(\mathcal{A}, \xi) = \{1, 2, ..., 6, 8, 10, 12, ...\}$ can be used. The $(0, 5)$ RLL code discussed above cannot be optimal because its runlength set is a proper subset of $\mathcal{L}(\mathcal{A}, \xi)$. We denote the rate of a (d, k) RLL code by $C(d, k)$, and we have $C(0, 5) = 0.9881$ and $C(0, \infty) = 1$ [2, P.62]. The rate of the $[\mathcal{A}, \xi]$ code as prescribed should be somewhere in between and it is found to be 0.9914 by the results in the next section. When ξ is large, an optimal $[\mathcal{A}, \xi]$ code can perform much better than an RLL code, as to be shown next.

ii) Consider $a = 2$ and $\xi = 1.51$. Then we see that the $(1, 1)$ RLL code is the best possible RLL code for zero-error communication. The rate of this code is equal to 0 because there are only two codewords. Alternatively, an $[\mathcal{A}, \xi]$ code with $\mathcal{L}(\mathcal{A}, \xi) = \{2, 4, 7, 11, ...\}$ can be used, and the rate is 0.4383. In this example, the gain from not imposing an upper limit on the runlengths is infinite!

Since Ψ is an optimal code in $\mathcal{C}_L$, it can be used to define the capacity of the $[\mathcal{A}, \xi]$ channel. The capacity, however, can be easily determined if we do it in another way. We define the subclass of $\mathcal{C}_L$, denoted by $\mathcal{C}_E$, such that all the step functions in $\mathcal{C}_E$ have duration $t = T$ and the lengths of the runs, except for possibly the last run, take values in the list $\mathcal{L}(\mathcal{A}, \xi)$. The only difference between $\mathcal{C}_E$ and $\mathcal{C}_L$ arises from Points 1 and 3. Let M_E be the number of codewords in the optimal code in $\mathcal{C}_E$ and let M_L be the number of codewords in Ψ. It can be shown that

$$\lim_{T \to \infty} \frac{1}{T} \log M_L = \lim_{T \to \infty} \frac{1}{T} \log M_E.$$

Therefore, both definitions of capacity are the same in the asymptotic sense.

Definition 1: The capacity of the $[\mathcal{A}, \xi]$ channel is defined as $C[p, \mathcal{L}(\mathcal{A}, \xi)] = \lim_{T \to \infty} \frac{1}{T} \log M_E$ if it exists.

The capacity $C[p, \mathcal{L}]$ can be given in terms of the characteristic function $Q(\mathcal{L}, z)$ which is defined as

$$Q(\mathcal{L}, z) = 1 - (p-1) \sum_{l \in \mathcal{L}} z^l. \tag{1}$$

If (1) converges for some z, let $\mu(\mathcal{L})$ be the unique positive real root of (1). The capacity $C[p, \mathcal{L}(\mathcal{A}, \xi)]$ is shown [1, Theorem 2] to be

$$C[p, \mathcal{L}(\mathcal{A}, \xi)] = -\log \mu(\mathcal{L}). \tag{2}$$

When $p = 2$, the above result is reduced to Proposition 1.1 in [3]. If $\mathcal{L}$ is a subset of integers, then $Q(\mathcal{L}, z)$ converges on $[0, \theta]$ for all $\theta \in (0, 1)$ and $-\log \mu(\mathcal{L}) \leq \log p$.

In general, $Q(\mathcal{L}, z)$ may not have a closed form so that $\mu(\mathcal{L})$ is difficult to compute. In this case, we may need to approximate $\mu(\mathcal{L})$ by $\mu_m(\mathcal{L})$ for a sufficiently large m, where $\mu_m(\mathcal{L})$ is the unique positive root of

$$Q_m(\mathcal{L}, z) \quad = \quad 1 - (p-1) \sum_{l \in \mathcal{L}_m} z^l,$$

where $\mathcal{L}_m = \mathcal{L} \cap [0, m]$. Thus the convergence rate of $\{\mu_m(\mathcal{L})\}$ is of practical interest.

Theorem 1 (The Convergence Rate): For any $0 < \theta_1 < \theta_2$ with $Q(\mathcal{L}, \theta_1) \geq 0$ and $Q(\mathcal{L}, \theta_2) \leq 0$,

$$0 \leq \mu_m(\mathcal{L}) - \mu(\mathcal{L}) \leq \frac{\sum_{l \in \mathcal{L} \setminus \mathcal{L}_m} {\theta_2}^l}{\sum_{l \in \mathcal{L}'_m} l\mu_m(\mathcal{L})^{l-1} + \sum_{l \in \mathcal{L}''_m} l{\theta_1}^{l-1}}, \tag{3}$$

where $\mathcal{L}'_m = \{l \in \mathcal{L}_m : l \leq 1\}$ and $\mathcal{L}''_m = \{l \in \mathcal{L}_m : l > 1\}$.

Proof: Since $Q_m(\mathcal{L}, \cdot)$ is monotonically decreasing (for $z \geq 0$), its inverse function $Q_m^{-1}(\mathcal{L}, \cdot)$ exists and has derivative

$$\frac{dQ_m^{-1}(\mathcal{L}, u)}{du} = -\frac{1}{(p-1)\sum_{l \in \mathcal{L}_m} l(Q_m^{-1}(\mathcal{L}, u))^{l-1}}. \tag{4}$$

Moreover we have

$$Q_m^{-1}\left(\mathcal{L}, (p-1)\sum_{l \in \mathcal{L} \setminus \mathcal{L}_m} \mu(\mathcal{L})^l\right) = \mu(\mathcal{L}),$$

and

$$Q_m^{-1}(\mathcal{L}, 0) = \mu_m(\mathcal{L}).$$

Then by Lagrange's mean value theorem and (4), there exists an $\eta \in (0, (p-1)\sum_{l \in \mathcal{L} \setminus \mathcal{L}_m} \mu(\mathcal{L})^l)$ such that

$$\begin{aligned} 0 &\leq \mu_m(\mathcal{L}) - \mu(\mathcal{L}) \\ &= Q_m^{-1}(\mathcal{L}, 0) - Q_m^{-1}\left(\mathcal{L}, (p-1)\sum_{l \in \mathcal{L} \setminus \mathcal{L}_m} \mu(\mathcal{L})^l\right) \\ &= \frac{\sum_{l \in \mathcal{L} \setminus \mathcal{L}_m} \mu(\mathcal{L})^l}{\sum_{l \in \mathcal{L}_m} l(Q_m^{-1}(\mathcal{L}, \eta))^{l-1}}. \end{aligned} \tag{5}$$

By the monotonicity of $Q_m^{-1}(\mathcal{L}, \cdot)$, we have

$$\mu(\mathcal{L}) \leq Q_m^{-1}(\mathcal{L}, \eta) \leq \mu_m(\mathcal{L}).$$

In the course of lower bounding the denominator in (5), we have to distinguish two cases. From the inequality above, for $l \in \mathcal{L}'_m$, i.e., $l \leq 1$ (so that $l - 1 \leq 0$), we have

$$(Q_m^{-1}(\mathcal{L}, \eta))^{l-1} \geq \mu_m(\mathcal{L})^{l-1},$$

and for $l \in \mathcal{L}''_m$, i.e., $l > 1$, we have

$$(Q_m^{-1}(\mathcal{L}, \eta))^{l-1} \geq \mu(\mathcal{L})^{l-1}.$$

Thus

$$\sum_{l \in \mathcal{L}_m} l(Q_m^{-1}(\mathcal{L}, \eta))^{l-1} \geq \sum_{l \in \mathcal{L}'_m} l\mu_m(\mathcal{L})^{l-1} + \sum_{l \in \mathcal{L}''_m} l\mu(\mathcal{L})^{l-1}.$$

Then the theorem can be justified because $Q(\mathcal{L}, \theta_1) \geq 0$ and $Q(\mathcal{L}, \theta_2) \leq 0$ imply

$$\theta_1 \leq \mu(\mathcal{L}) \leq \theta_2.$$

■

In addition to the convergence rate, this theorem has the following physical interpretation. Note that the transmitter and receiver clocks can always re-synchronize with each other at the end of a runlength. This means that when the set of runlengths is $\mathcal{L}_m$, the transmitter and receiver clocks can always re-synchronize with each other within a horizon of length m. Thus if we impose the constraint that the transmitter and receiver clocks cannot be out-of-sync for more than m time units, the rate loss incurred is at most

$$-\log \mu(\mathcal{L}) + \log \mu(\mathcal{L}_m),$$

which by (3) is upper bounded by

$$-\log(\mu(\mathcal{L}_m) - \delta_m) + \log \mu(\mathcal{L}_m),$$

where δ_m denotes the RHS of (3) and we have invoked the monotonicity of the logarithmic function.

Corollary 1: If $\mathcal{L}$ is a subset of positive integers and $Q(\mathcal{L}, \theta) \leq 0$, then

$$\begin{aligned} 0 &\leq \mu_m(\mathcal{L}) - \mu(\mathcal{L}) \\ &\leq \frac{\theta^{m+1}}{(1-\theta)\left(\sum_{l \in \mathcal{L}'_m} l\mu_m(\mathcal{L})^{l-1} + \tilde{d}p^{-(\tilde{d}-1)}\right)}, \end{aligned}$$

where $\tilde{d}$ is the smallest element of $\mathcal{L}''_m$.

III. The Probabilistic Capacity

The Maxentropic Theorem for Markov information sources, which is an important result in information theory, can go back to Shannon's original work [4], and the Maxentropic Theorem for (dk) sequences can be obtained as its special case (cf. [2] for a detailed discussion and further references). For our problem, when the list $\mathcal{L}$ of runlengths is an infinite set, the information source has infinite memory and therefore is not Markovian. So the approach for proving the Maxentropic Theorem via transition matrices does not work. In this section, the theorem will be obtained by way of the characteristic function $Q(\cdot, \cdot)$. The theorem asserts that the probabilistic capacity and the combinatorial capacity are the same for $\mathcal{L}$ being an infinite set of runlengths.

Definition 2: For a given list $\mathcal{L}$ of runlengths and an alphabet $\mathcal{X}$ with cardinality p, let $\mathcal{S}(\mathcal{L})$ be the set of step functions on $[0, \infty)$ taking values in $\mathcal{X}$ whose runlengths are in $\mathcal{L}$, $\mathcal{K}(\mathcal{L})$ be the set of stochastic processes $X(t)$ on $[0, \infty)$ taking values almost surely (a.s.) in $\mathcal{S}(\mathcal{L})$, $X^T(t) := \{X(t) : 0 \leq t \leq T\}$ be the Tth segment of stochastic process $X(t)$, and $P_{X(t)}$ and $P_{X^T(t)}$ be the distributions of $X(t)$ and $X^T(t)$, respectively. Then the entropy rate of $X(t) \in \mathcal{K}(\mathcal{L})$ is defined by

$$r(X(t)) := \lim_{T \to \infty} -\frac{1}{T}\mathbf{E} \log P_{X^T(t)}(X^T(t)) \tag{6}$$

and the maxentropy of $\mathcal{K}(\mathcal{L})$ is defined by

$$r(\mathcal{L}) := \max_{X(t) \in \mathcal{K}(\mathcal{L})} r(X(t)). \tag{7}$$

In order for the maxentropy to be well defined, we have to show that the maximum in (7) is achievable. Under a reasonable condition, this will be done in the proof of the next theorem. Actually, we will show that the stochastic process $X^*(t) \in \mathcal{K}(\mathcal{L})$ defined below achieves the maxentropy.

Any step function $S(t) \in \mathcal{S}(\mathcal{L})$ can be uniquely determined by a pair of sequences, $l^\infty = (l_1, l_2, l_3, \ldots)$ and $y^\infty = (y_1, y_2, y_3, \ldots)$, where $l_i \in \mathcal{L}$ and $y_j \in \mathcal{X}$ are respectively the ith runlength and the value taken by the jth run of $S(t)$. We will write $\Im(S(t)) = (l^\infty, y^\infty)$ and $\Im^{-1}(l^\infty, y^\infty) = S(t)$. Similarly, a stochastic process $X(t) \in \mathcal{K}(\mathcal{L})$ is uniquely determined by a pair of stochastic sequences $L(i)$ and $Y(j)$. Likewise, $\Im(X(t)) = (L(i), Y(j))$ and $\Im^{-1}(L(i), Y(j)) = X(t)$. This allows us to define $X^*(t)$ by defining $\Im(X^*(t))$. For a list $\mathcal{L}$, $\mu(\mathcal{L})$ satisfies $1 - (p-1)\sum_{l\in\mathcal{L}}(\mu(\mathcal{L}))^l = Q(\mathcal{L}, \mu(\mathcal{L})) = 0$, and so $\sum_{l\in\mathcal{L}}(p-1)(\mu(\mathcal{L}))^l = 1$. Thus $P_{\mu(\mathcal{L})} := \{P_{\mu(\mathcal{L})}(l) = (p-1)(\mu(\mathcal{L}))^l : l \in \mathcal{L}\}$ is a probability distribution over $\mathcal{L}$. Let $\{L^*(i)\}_{i=1}^\infty$ be an i.i.d. stochastic sequence with distribution $P_{\mu(\mathcal{L})}$ and $\{Y^*(j)\}_{j=1}^\infty$ be a Markov chain which is independent of $\{L^*(i)\}_{i=1}^\infty$ and has the following distribution, where a is any fixed letter in $\mathcal{X}$.

$$Pr(Y^*(1) = x) = \begin{cases} 0 & \text{if } x = a \\ \frac{1}{p-1} & \text{otherwise,} \end{cases} \tag{8}$$

and for $j = 2, 3, \ldots,$

$$Pr(Y^*(j) = x_j | Y^*(j-1) = x_{j-1}) = \begin{cases} 0 & \text{if } x_j = x_{j-1} \\ \frac{1}{p-1} & \text{otherwise.} \end{cases} \tag{9}$$

Define $X^*(t) = \Im^{-1}(L^*(i), Y^*(j))$. Then $X^*(t) \in \mathcal{K}(\mathcal{L})$. Notice that if instead of (8) we let $Y^*(1)$ be uniformly distributed over $\mathcal{X}$, then the entropy would be increased, but for the entropy rate in (6) which we are interested in, it does not make any difference.

Theorem 2 (Maxentropic Theorem): For any list $\mathcal{L}$, alphabet $\mathcal{X}$ of cardinality p, and any $X(t) \in \mathcal{K}(\mathcal{L})$,

$$r(X(t)) \le -\log \mu(\mathcal{L}). \tag{10}$$

In the above, equality is achieved by $X^*(t)$, and hence

$$r(\mathcal{L}) = -log\mu(\mathcal{L}). \tag{11}$$

Proof: For the upper bound in (10), we have

$$\begin{aligned} r(X(t)) &= \lim_{T\to\infty} -\frac{1}{T}\mathbf{E}\log P_{X^T(t)}(X^T(t)) \\ &= \lim_{T\to\infty} \frac{1}{T} H(X^T(t)) \\ &\le \lim_{T\to\infty} \frac{1}{T}\log N(T) \\ &= -\log \mu(\mathcal{L}), \end{aligned}$$

where $N(t)$ is the total number of possible $X^T(t)$ and the last equality follows from (2). The proof of (11) is shown in [5] which is too long to appear here. The results in the next section, however, can support that $X^*(t)$ can achieve the maximum entropy. ∎

IV. The Interval Code

The relation between coding in constrained channels and random number generation is shown in this section. The stochastic process $X^*(t)$ described in the last section is used as an input to the constrained channel. By assuming the information source generates a sequence of fair bits, the encoding process is equivalent to converting a stream of fair bits into the stochastic process $X^*(t)$. On the other hand, the decoding process is just the reverse, i.e., converting the stochastic process $X^*(t)$ into fair bits. We call this code as *interval code* because our proposed encoding and decoding algorithms are modified from the interval algorithm [6] which converts any two stochastic processes from one to another. Due to Theorem 2 and the optimality of the interval algorithm, interval code is asymptotically optimal. Moreover, the interval code can still operate for $\mathcal{L}$ being a countably infinite set of runlengths which cannot be done by the state-splitting algorithm [7][8].

We now present the encoding and decoding algorithms for the interval code. Define a random variable $\hat{Z}$ which takes value from the set

$$\{(y, i) : y \in [1, p-1] \text{ and } i \in [1, |\mathcal{L}|]\}$$

and $\hat{Z} = (y, i)$ with probability $(\mu(\mathcal{L}))^{l_i^*}$. Let $f(y, i) = (i-1)(p-1) + y$ and define a random variable $Z = f(\hat{Z})$. Since f is bijective, Z and $\hat{Z}$ share the same distribution. For simplicity, we denote the probability distribution of Z by $\mathcal{Q} = \{q_1, q_2, \ldots\}$ and $\Pr\{Z = z\} = q_z$. For a larger i, l_i^* is larger and $(\mu(\mathcal{L}))^{l_i^*}$ is smaller as $0 < \mu(\mathcal{L}) < 1$. It is readily checked that q_z's are in non-increasing order. If $\mathcal{L}$ has a countably infinite number of runlengths, the cardinality of $\mathcal{Q}$ is also countably infinite. We use $\mathbf{Z}$ to denote a random vector $(Z_1, Z_2, \ldots, Z_N)$, where Z_i's are i.i.d. with the same distribution $\mathcal{Q}$. Then it is easily seen that a segment of the stochastic process $X^*(t)$ can be uniquely determined by $\mathbf{Z}$, and vice versa for the determination of $\mathbf{Z}$ from a segment of $X^*(t)$. We temporarily assume the information source is generating fair bits which will be generalized at the end of this section. We write $\mathbf{U} = (U_1, U_2, \ldots, U_M)$ as a vector of fair bits. We call the encoding algorithm "Interval Encoding" because it is basically a successive refinement of interval partitions. Since $\mathcal{Q}$ can have a countably infinite number of probability masses, the interval encoding is slightly different from the interval algorithm in [6]. The only main difference comes from the end of the encoding process. The following algorithm encodes a random vector $\mathbf{U} = (U_1, U_2, \ldots, U_M)$ into a random vector $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_N)$.

Interval Encoding:

1) Let $P_0 = 0.5$ and $P_1 = 1$.
2) Initialize $c = 1$, $n = 1$ and $\mathbf{U} = \mathbf{Z} = (\emptyset)$, i.e., an empty vector. Let $\alpha_{\mathbf{U}} = \alpha_{\mathbf{Z}} = 0$, $\beta_{\mathbf{U}} = \beta_{\mathbf{Z}} = 1$, $I(\mathbf{U}) = [\alpha_{\mathbf{U}}, \beta_{\mathbf{U}})$, and $J(\mathbf{Z}) = [\alpha_{\mathbf{Z}}, \beta_{\mathbf{Z}})$.
3) If $c = M + 1$, then go to 6). Otherwise, let u be the c-th input bit, i.e., the realization of U_c. Set $c = c + 1$ and $\mathbf{U}' = (\mathbf{U}, u)$. Then the subinterval of $I(\mathbf{U})$ is generated as

$$I(\mathbf{U}') = [\alpha_{\mathbf{U}'}, \beta_{\mathbf{U}'}),$$

where

$$\alpha_{\mathbf{U}'} = \alpha_{\mathbf{U}} + (\beta_{\mathbf{U}} - \alpha_{\mathbf{U}})(P_u - 0.5)$$

and

$$\beta_{\mathbf{U}'} = \alpha_{\mathbf{U}} + (\beta_{\mathbf{U}} - \alpha_{\mathbf{U}})P_u.$$

4a) Let j be the largest nonnegative integer such that

$$\alpha_{\mathbf{Z}} + (\beta_{\mathbf{Z}} - \alpha_{\mathbf{Z}}) \sum_{i=1}^{j} q_i \leq \alpha_{\mathbf{U}'}.$$

// Note that $j = 0$ if $\alpha_{\mathbf{Z}} + (\beta_{\mathbf{Z}} - \alpha_{\mathbf{Z}})q_1 > \alpha_{\mathbf{U}'}$. Then the subinterval of $J(\mathbf{Z})$ is generated as

$$J(\mathbf{Z}') = [\alpha_{\mathbf{Z}'}, \beta_{\mathbf{Z}'}),$$

where

$$\alpha_{\mathbf{Z}'} = \alpha_{\mathbf{Z}} + (\beta_{\mathbf{Z}} - \alpha_{\mathbf{Z}}) \sum_{i=1}^{j} q_i$$

and

$$\beta_{\mathbf{Z}'} = \alpha_{\mathbf{Z}} + (\beta_{\mathbf{Z}} - \alpha_{\mathbf{Z}}) \sum_{i=1}^{j+1} q_i.$$

If $I(\mathbf{U}') \nsubseteq J(\mathbf{Z}')$, then go to 5).

4b) Set $Z_n = j+1$, i.e., the n-th output and set $\mathbf{Z} = (\mathbf{Z}, Z_n)$. Update $\alpha_{\mathbf{Z}} = \alpha_{\mathbf{Z}'}$ and $\beta_{\mathbf{Z}} = \beta_{\mathbf{Z}'}$. Set $n = n + 1$ and go to 4a).

5) Set $\mathbf{U} = \mathbf{U}'$, $\alpha_{\mathbf{U}} = \alpha_{\mathbf{U}'}$, $\beta_{\mathbf{U}} = \beta_{\mathbf{U}'}$ and go back to 3).

6) Set $Z_n = j+1$, i.e., the last output and set $\mathbf{Z} = (\mathbf{Z}, Z_n)$. The algorithm ends here.

The following "Interval Decoding" is basically the reverse of the interval encoding. It can also be interpreted as an encoding of $\mathbf{Z}$ into $\mathbf{U}$ except that the last run is handled in a different way. The following algorithm decodes a random vector $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_N)$ to a random vector $\mathbf{U} = (U_1, U_2, \ldots, U_M)$. We temporarily assume the values of M and N are known at the decoder side which will be generalized at the end of this section.

Interval Decoding:

1) Let $P_0 = 0.5$ and $P_1 = 1$.

2) Initialize $c = 1$, $m = 1$ and $\mathbf{U} = \mathbf{Z} = (\emptyset)$, i.e., an empty vector. Let $\alpha_{\mathbf{U}} = \alpha_{\mathbf{Z}} = 0$, $\beta_{\mathbf{U}} = \beta_{\mathbf{Z}} = 1$, $I(\mathbf{U}) = [\alpha_{\mathbf{U}}, \beta_{\mathbf{U}})$, and $J(\mathbf{Z}) = [\alpha_{\mathbf{Z}}, \beta_{\mathbf{Z}})$.

3) If $c = N+1$, then go to 6a). Otherwise, let z be the c-th input symbol, i.e., the realization of Z_c. Set $c = c+1$ and $\mathbf{Z}' = (\mathbf{Z}, z)$. Then the subinterval of $J(\mathbf{Z})$ is generated as

$$J(\mathbf{Z}') = [\alpha_{\mathbf{Z}'}, \beta_{\mathbf{Z}'}),$$

where

$$\alpha_{\mathbf{Z}'} = \alpha_{\mathbf{Z}} + (\beta_{\mathbf{Z}} - \alpha_{\mathbf{Z}}) \sum_{i=1}^{z-1} q_i$$

and

$$\beta_{\mathbf{Z}'} = \alpha_{\mathbf{Z}} + (\beta_{\mathbf{Z}} - \alpha_{\mathbf{Z}}) \sum_{i=1}^{z} q_i.$$

4a) If $\alpha_{\mathbf{U}} + 0.5(\beta_{\mathbf{U}} - \alpha_{\mathbf{U}}) \leq \alpha_{\mathbf{Z}'}$, let $j = 1$. Otherwise, let $j = 0$. Then the subinterval of $I(\mathbf{U})$ is generated as

$$I(\mathbf{U}') = [\alpha_{\mathbf{U}'}, \beta_{\mathbf{U}'}),$$

where

$$\alpha_{\mathbf{U}'} = \alpha_{\mathbf{U}} + (\beta_{\mathbf{U}} - \alpha_{\mathbf{U}})(P_j - 0.5)$$

and

$$\beta_{\mathbf{U}'} = \alpha_{\mathbf{U}} + (\beta_{\mathbf{U}} - \alpha_{\mathbf{U}})P_j.$$

If $J(\mathbf{Z}') \nsubseteq I(\mathbf{U}')$, then go to 5).

4b) Set $U_m = j$, i.e., the m-th output bit and set $\mathbf{U} = (\mathbf{U}, U_m)$. Update $\alpha_{\mathbf{U}} = \alpha_{\mathbf{U}'}$ and $\beta_{\mathbf{U}} = \beta_{\mathbf{U}'}$. Set $m = m + 1$ and go to 4a).

5) Set $\mathbf{Z} = \mathbf{Z}'$, $\alpha_{\mathbf{Z}} = \alpha_{\mathbf{Z}'}$, $\beta_{\mathbf{Z}} = \beta_{\mathbf{Z}'}$ and go back to 3).

6a) If $m = M + 1$, then go to 7). Otherwise, If

$$\alpha_{\mathbf{U}} + 0.5(\beta_{\mathbf{U}} - \alpha_{\mathbf{U}}) \leq \beta_{\mathbf{Z}'},$$

let $j = 1$. Otherwise, let $j = 0$. Then the subinterval of $I(\mathbf{U})$ is generated as

$$I(\mathbf{U}') = [\alpha_{\mathbf{U}'}, \beta_{\mathbf{U}'}),$$

where

$$\alpha_{\mathbf{U}'} = \alpha_{\mathbf{U}} + (\beta_{\mathbf{U}} - \alpha_{\mathbf{U}})(P_j - 0.5)$$

and

$$\beta_{\mathbf{U}'} = \alpha_{\mathbf{U}} + (\beta_{\mathbf{U}} - \alpha_{\mathbf{U}})P_j.$$

6b) Set $U_m = j$, i.e., the m-th output bit and set $\mathbf{U} = (\mathbf{U}, U_m)$. Update $\alpha_{\mathbf{U}} = \alpha_{\mathbf{U}'}$ and $\beta_{\mathbf{U}} = \beta_{\mathbf{U}'}$. Set $m = m + 1$ and go to 6a).

7) The algorithm ends here.

Since we are encoding a binary input, the conversion from $\mathbf{Z}$ to $\mathbf{U}$ is similar to the encoding process of the arithmetic code while the conversion from $\mathbf{U}$ to $\mathbf{Z}$ is similar to the decoding process of the arithmetic code. The main difference comes from the handling of the last input random variable. Moreover, both the encoding and decoding algorithms can progressively produce output without waiting for the end of the input random vector.

We now consider the asymptotic performance of the interval code. Assume that the source generates an infinite number of U_i's, i.e., $M = \infty$, and the destination decodes an infinite number of Z_i's, i.e., $N = \infty$. Then Step 6 in the interval encoding and Steps 6-7 in the interval decoding will never be reached, and the interval encoding and decoding are exactly the same as the interval algorithm for random number generation. Therefore, by applying the results in [6] here, we have

$$NH(Z) \leq \mathbf{E}[M] \leq NH(Z) + 3, \tag{12}$$

where the inequalities follow from [6, Theorem 1] and [6, eq. (5.1)], respectively. Together with

$$\begin{aligned} H(Z) &= -\sum_{x=1}^{p-1}\sum_{l\in\mathcal{L}}(\mu(\mathcal{L}))^l \log(\mu(\mathcal{L}))^l \\ &= (-\log\mu(\mathcal{L}))\cdot(p-1)\sum_{l\in\mathcal{L}} l(\mu(\mathcal{L}))^l, \end{aligned}$$

we have bounded the expected number of fair bits, $\mathbf{E}[M]$, in (12) for generating N runs. Since $\hat{Z} = f^{-1}(Z) = (y, i)$ and a runlength $l_i^* \in \mathcal{L}$ is produced with probability $(\mu(\mathcal{L}))^{l_i^*}$, the expected length of a run is given by

$$\sum_{x=1}^{p-1}\sum_{l\in\mathcal{L}} l(\mu(\mathcal{L}))^l = (p-1)\sum_{l} l(\mu(\mathcal{L}))^l. \tag{13}$$

Let T be the duration of a communication session. For a large T, we can apply the law of large number to approximate the number of runs sent through the channel by

$$N \approx \frac{T}{(p-1)\sum_l l(\mu(\mathcal{L}))^l} = \frac{T(-\log\mu)}{H(Z)}.$$

Together with (12), we have

$$-\log\mu \le \frac{\mathbf{E}[M]}{T} \le -\log\mu + \frac{3}{T},$$

and

$$\lim_{T\to\infty}\frac{\mathbf{E}[M]}{T} = -\log\mu.$$

Therefore, the interval code is asymptotically optimal. This result can also give an alternative proof to (11).

Now, we consider a finite communication session, i.e., M is finite. Denote the size of an interval $I = [\alpha, \beta)$ by $|I|$ which is equal to $\beta - \alpha$ in this case. When Step 6 in the interval encoding is just reached, $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_{N-1})$ and

$$|J(\mathbf{Z})| \ge |I(\mathbf{U})| = 2^{-M}, \tag{14}$$

from which the duration of a step function converted from $\mathbf{Z}$ can be determined. Let $(y_i, \lambda_i) = f^{-1}(Z_i)$ for $1 \le i \le N-1$. Then the i-th runlength of the step function is $l_i = l^*_{\lambda_i}$, so that

$$|J(\mathbf{Z})| = \prod_{i=1}^{N-1}(\mu(\mathcal{L}))^{l_i}.$$

The length of the step function converted from $\mathbf{Z}$ is given by

$$\sum_{i=1}^{N-1} l_i = \frac{\log|J(\mathbf{Z})|}{\log\mu(\mathcal{L})} \le \frac{M}{-\log\mu(\mathcal{L})},$$

where the inequality follows from (14) and $\log\mu(\mathcal{L}) < 0$. Finally, the Z_N is generated in Step 6 by finding an interval $J(\mathbf{Z}')$ such that $\alpha_{\mathbf{U}'} \in J(\mathbf{Z}')$. This is a biased random number generation and therefore, $\mathbf{E}[Z_N] \le \mathbf{E}[Z]$. Then the expected runlength converted from Z_N is smaller than (13). Therefore, the total length of the step function is upper bounded by

$$\frac{M}{-\log\mu(\mathcal{L})} + (p-1)\sum_l l(\mu(\mathcal{L}))^l,$$

from which we see that the overhead of the interval code is only one extra run in the step function.

There are many variety of the interval code for different applications. Suppose the decoder is required to automatically determine the value of M and N through the received step function. If M is known at the beginning, we may encode M into a header, for example, by the Elias code [9] and send the header before sending the random vector $\mathbf{U}$. The length of the header is upper bounded by $2\log M + 1$ so that the length of the header is negligible when M is large. Accordingly, Step 6 in the interval encoding is changed to finding the largest subinterval $J(\mathbf{Z}')$ such that $J(\mathbf{Z}') \subset I(\mathbf{U}')$. Then the encoder outputs the random variables in $\mathbf{Z}'$ but not in $\mathbf{Z}$ before the algorithm ends. At the decoder side, the header together with $\mathbf{U}$ can be uniquely decoded due to $J(\mathbf{Z}') \subset I(\mathbf{U}')$ so that the decoder can determine M at the beginning. Then the first thing to do in Step 3 in the interval decoding is changed to "if $m = M + 1$, then the algorithm ends" instead of checking the value of c.

We have seen the close relation between the interval code and the arithmetic code. If the information source is not generating fair bits, we may combine the interval code with the arithmetic code. The trick is to divide the interval $I(\mathbf{U})$ according to the source input distribution which may be any stochastic process. Then a simple joint source-channel coding for a constrained channel can be obtained.

There are some possible ways to modify the interval code to cope with possible errors in the received step function. For example, we may use an error-correcting code to protect the data before the interval encoding is used. The possibility of deriving an error-correcting constrained coding from the interval code will also be an interesting future work. After all, how to incorporate both time jitter and noise into the same model while keeping the model analytically tractable is perhaps the biggest challenge for future research in this direction.

REFERENCES

[1] N. Cai and R. W. Yeung, "Self-Synchronizable Codes for Asynchronous Communication", in *Proc. 2002 IEEE Int. Symposium Inform. Theory (ISIT 2002)*, Lausanne, Switzerland, June 30-July 5, 2002.

[2] K. A. S. Immink, *Codes for Mass Data Storage Systems*, Shannon Foundation Publishers, The Netherlands, 1999.

[3] I. Csiszár, "Simple Proofs of Some Theorems on Noiseless Channels", in *Information and Control*, V. 14, pp. 285-298, 1969.

[4] C. E. Shannon, A Mathematical Theory of Communiction, *Bell Tech. J.*, V. 27, pp. 379-423, July 1948.

[5] R. W. Yeung, N. Cai and S.-W. Ho, "Reliable Communication in the Absence of a Common Clock", submitted to *IEEE Trans. Inform. Theory*.

[6] T. S. Han and M Hoshi, "Interval Algorithm for Random Number Generation", in *IEEE Trans. Inform. Theory*, IT-43: 599-611, 1997.

[7] R. L. Alder, D. Coppersmith and M. Hassner, " Algorithms for Sliding Block Codes: An Application of Symbolic Dynamics to Information Theory", in *IEEE Trans. Inform. Theory*, IT-29: 5-22, 1983.

[8] K. A. S. Immink, P. H. Siegel and J. K. Wolf, " Codes for Digital Recorders", in *IEEE Trans. Inform. Theory*, IT-44: 2260-2299, 1998.

[9] P. Elias, " Universal Codeword Sets and Representations of the Integers", in *IEEE Trans. Inform. Theory*, IT-21: 194-203, 1975.